Gray-box Techniques for Adversarial Text Generation

Prithviraj Dasgupta, Joseph Collins and Anna Buhman[*]

Abstract

We consider the problem of adversarial text generation in the context of cyber-security tasks such as email spam filtering and text classification for sentiment analysis on social media sites. In adversarial text generation, an adversary attempts to perturb valid text data to generate adversarial text such that the adversarial text ends up getting mis-classified by a machine classifier. Many existing techniques for perturbing text data use gradient-based or white-box methods, where the adversary observes the gradient of the loss function from the classifier for a given input sample, and uses this information to strategically select portions of the text to perturb. On the other hand, black-box methods where the adversary does not have access to the gradient of the loss function from the classifier and has to probe the classifier with different input samples to generate successful adversarial samples, have been used less often for generating adversarial text. In this paper, we integrate black-box methods where the adversary has a limited budget of the number of probes to the classifier, with white-box, gradient-based methods, and evaluate the effectiveness of the adversarially generated text in misleading a deep network classifier model.

Introduction

Machine learning-based systems are currently used widely for different cyber-security tasks including email spam filtering, network intrusion detection, sentiment analysis of posts on social media sites like Twitter, and validating authenticity of news posts on social media sites. Most of these systems use supervised learning techniques where a learning algorithm called a classifier is trained to categorize data into multiple categories. For instance, an automated classifier for an email spam filter classifies incoming email into spam and non-spam categories. A critical vulnerability of classifier-based learning algorithm is that they are susceptible to adversarial attacks where a malicious adversary could send corrupted or poisoned data to the classifier that results in incorrect classification, or, for more calculated attacks, use corrupted data to alter the classification decisions of the classifier. Both these attacks could compromise the operation of the classifier, making it behave in an unintended, possibly insecure and unsafe manner. As an example, a compromised email classifier could end up categorizing valid email messages as spam (false positives) while delivering spam email messages as non-spam (false negatives). An important defense against adversarial attacks is to build a model of the adversary including the data that the adversary generates, and use this model as a virtual adversary to improve the learner's robustness to adversarial attacks. Towards this objective, in this paper, we investigate adversarial data generation techniques that could be used to model corrupted data generated by a virtual adversary. Our work focuses on adversarial text data generation as many of the aforementioned cyber-security tasks operate mainly on text data.

Adversarial data generation techniques can be broadly classified into two categories called white-box and black-box. In white-box techniques, the adversary has access to information of the classifier, such as the parameters of the model of the classifier, e.g., weights in a deep neural network-based classifier, or, to internal calculations of the classifier, such as the gradients of the loss function calculated by the classifier on a sample. In contrast, in black-box techniques, the adversary does not have any information about the classifier and treats it like a black-box, by passing data samples as queries to the classifier and observing the classifier's output category or label for each sample. Additionally, in most practical black-box settings, the adversary can send only a finite number of queries to the classifier due to the adversary's budget limitations in generating samples, or, due to restrictions in the number of queries accepted by the classifier. While white-box techniques such as gradient-based methods (Liang et al. 2018) have been proposed for generating adversarial text, black box methods for generating adversarial text (Gao et al. 2018) are less investigated in literature. In this paper, we have describe gray-box techniques for generating adversarial text where gradient-based white-box techniques for generating adversarial text are combined with budget-limited, black-box techniques for

[*]P. Dasgupta and A. Buhman are with the Computer Science Dept., University of Nebraska at Omaha, USA. e-mail: {pdasgupta, abuhman}@unomaha.edu. J. Collins is with the Distributed Systems Section, Naval Research Laboratory, Washington D.C., USA. e-mail: joseph.collins@nrl.navy.mil

generating adversarial samples. We have evaluated the adversarial text data generated using our proposed technique using the DBPedia dataset that contains short articles on 14 different categories extracted from Wikipedia, in terms of the difference from the original data used as a basis to generate the adversarial data, as well as the effectiveness of the adversarial data in fooling a classifier into making misclassifications. Our results show that using gray-box techniques the divergence in the perturbed text from the original, unperturbed text increases with the amount of perturbation, and, more perturbation of original text results in changing the label of the perturbed text with respect to the original text more often.

Related Work

Adversarial data generation techniques have received considerable attention from researchers in the recent past. The main concept in adversarial data generation is to add slight noise to valid data using a perturbation technique so that the corrupted data still appears legitimate to a human but gets mis-classified by a machine classifier. Most research on adversarial data generation has focused on image data, including gradient-based methods for perturbing valid data (Biggio et al. 2013), (Goodfellow, Shlens, and Szegedy 2014), (Papernot et al. 2016), recurrent neural networks (Gregor et al. 2015), and generative adversarial networks (GANs) (Goodfellow et al. 2014) and its extensions (Mirza and Osindero 2014), (Chen et al. 2016), (Arjovsky, Chintala, and Bottou 2017). However, rather than generating synthetic data towards misleading a machine classifier, the objective of GAN-enabled data generation has been to create synthetic data that looks convincing to humans.

Adversarial Text Generation. For adverarial image generation, image characteristics like RGB pixel values, HSV values, brightness, contrast, etc., are real-valued numbers that can be manipulated using mathematical operations to generate perturbed images that can fool a machine classifer while remaining imperceptible to humans. In contrast, for perturbing text, adding a real value to a numerical representation of a word, character, or token in an embedding space, might result in a new word that might either be nonsense, or, might not fit in with the context of the unperturbed, original text - in both cases, the adversarial text can be easily flagged by both a machine classifier and a human. To address this shortcoming, instead of directly using image perturbation techniques to generate adversarial text, researchers have proposed using auto-encoders (Bengio et al. 2013) and recurrent neural networks (RNN) with long short term memory (LSTM) architecture to generate text as sequences of tokens (Mikolov et al. 2010), albeit with limitations when generating adversarial text that looks realistic to humans (Bengio et al. 2015), (Huszár 2015). Recently, GAN-based methods have been shown to be successful for adversarial text generation. In adverarial text generation GANs, a feedback signal (Yu et al. 2017), (Zhang et al. 2017) (Subramanian et al. 2017) such as a reward value within a reinforcement learning framework (Fedus, Goodfellow, and Dai 2018) or high-level features of text identified by the classifier called leaked information (Guo et al. 2017), is evaluated by the discriminator or classifier and provided to the generator or adversary. The adversary treats this information as a feedback signal from the classifier about the quality of its adversarially generated text, and adapts future adversarial generations of words or phrases in the text towards improving the quality of its generated text. Additional methods for generating adversarial text are described in (Iyyer et al. 2018), (Jia and Liang 2017), (Shen et al. 2017). Following gradient-based techniques for image perturbation (Goodfellow, Shlens, and Szegedy 2014), (Ebrahimi et al. 2018), (Liang et al. 2018) have used gradient-based methods to identify tokens in text that have the highest gradient and consequently most influence in determining the text's label in the classifier. These tokens are then strategically modified so that the text with modified tokens results in a different label when classified, as compared to the original text's label. Both these methods methods require the gradient information for the unperturbed text when it is classified by the classifier to perturb the text. In contrast, in (Gao et al. 2018), Gao *et al.* use a similar idea, but instead of selecting words or characters to perturb based on classifier-calculated gradients, they first rank words in a piece of text to be perturbed based on a score assigned to the word's context, followed by changing the spelling of the highest ranked words. Their technique is relevant when the gradient information from the classifier might not be readily available to perturb text, for example, in an online classification service where the classifier can be accessed only as a black box. Sethi and Kantardzic (Sethi and Kantardzic 2018), also described black-box methods for generating adversarial numerical and text data while considering budget limitations of the adversary in generating adversarial data. Our work is complimentary to these gradient-based and black-box approaches as it compares the effectiveness of generating adversarial text by integrating gradient-based methods with concepts used in budget-limited black-box methods.

Adversarial Text Perturbation

One of the main functions of an adversary in an adversarial learning setting is to generate, and possibly mislabel, adversarial text to fool the classifier. Adversarial text is usually generated by starting from valid text and changing certain words or characters in it. This process is also referrred to to as *perturbing* text. We define adversarial text perturbation using the following formalization: Let V denote a vocabulary of English words and X denote a corpus of English text. $\{X_i\} \subset X$ denotes a dataset consisting of English text samples, where X_i denotes the i-th sample. Further, $X_i = (X_i^{(1)}, X_i^{(2)}, ..., X_i^{(W)})$, where $X_i^{(j)}$ denotes the j-th word in X_i and W is the maximum number of words of a text sample. Each word $X_i^{(j)}$ is represented by a feature vector $X_i^{(j)} = \mathbf{f}_{1:F}$ generated using some word embedding like word2Vec (Mikolov et al. 2013), Global Vector (GloVe) (Pennington, Socher, and Manning 2014), or fastText (Joulin et al. 2016), where F is the embedding space dimension. Each sample X_i belongs to a class with label $l \in L$, where L is a finite set of class labels. For notational convenience, we assume that $l = y(X_i)$, where $y : X \to L$ is a relation that ascertains ground truth label l for X_i. A classifier C is also used to determine a label of X_i, the classifier's output is given by $y_C : X \rightarrow L$. We say that the classifier classifies X_i correctly only when $y_C(X_i) = y(X_i)$. A valid example X_i is perturbed by altering a subset of words $\{X_i^{(j)}\} \in X_i$ using a perturbation strategy π. The perturbation strategy $\pi : X \rightarrow X$ modifies the j-th word $X_i^{(j)}$ to word $\tilde{X}_i^{(j)} \in V, 1 \leq j \leq W$. Let $n_\pi = |\{\tilde{X}_i^{(j)}\}|$ denote the number of words perturbed by the perturbation strategy π and $\tilde{X}_{i,n_\pi}$ denote text X_i after perturbing n_π words in it. Finally, let $\Delta : X \times X \rightarrow [0,1]$ denote a divergence measure between two pieces of text with $\Delta(X_i, X_i) = 0$. Within this formalization, the objective of the adversary is to determine a minimal perturbation n_π^* to X_i satisfying:

$$\begin{aligned} n_\pi^* &= \arg_{n_\pi} \min \Delta(X_i, \tilde{X}_{i,n_\pi}) \\ \text{s.t.} \quad & y_C(\tilde{X}_{i,n_\pi}) \neq y(X_i), \\ \text{and,} \quad & y_C(X_i) = y(X_i) \end{aligned} \tag{1}$$

The objective function in Equation 1 finds the number of words to perturb that gives the minimum divergence between the original and perturbed text, while the last two constraints respectively ensure that the classifier mis-classifies the perturbed text, $\tilde{X}_{i,n_\pi}$, giving it a different label than the ground truth $y(X_i)$, but the classifier correctly classifies the original text X_i. In the next section, we describe different perturbation strategies we have used to perturb valid text and generate adversarial text.

Perturbation Methods for Text Data

An adversarial perturbation method adds a certain amount of noise to unperturbed data to generate perturbed data. There are two main questions in a perturbation method: how much noise to add, and where in the data to add the noise. For the first question, the two types of perturbation methods, white-box and black-box, both add a random, albeit small amount of noise. But these two types of methods differ in their approach to answer the second question. Below, we describe the white-box and black-box perturbation methods, and propose a gray-box method that combines the advantages of the two.

White-box Gradient-based Perturbation. White-box methods first query the classifier with the unperturbed text and observe the gradient of the loss function of the classifier's output from the query for the different words in the queried text. Because the loss function expresses the difference of the classifier's determined label from the ground truth label for the queried text, the word in the text that corresponds to the most negative gradient or change of the loss function is most influential in changing the label for the text determined by the classifier. This word is selected as the word to perturb. Mathematically, the selected word, x^*, having the most negative gradient is given by:

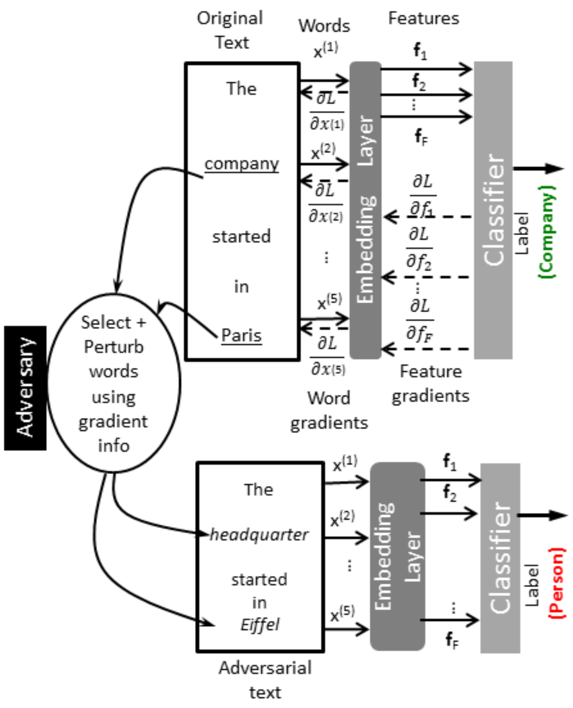


Figure 1: Example of white-box adversarial text generation via perturbing words using gradient information. (Top): The gradient of the loss function of the classifier from classifying original text is available to the adversary. (Left): Advery uses this information to select and perturb two words from original text. (Bottom): The perturbed text, when classified, yields a different label ('Person') than original text ('Company').

$$\begin{aligned} x^* &= \arg\min_j \frac{\delta L}{\delta x^{(j)}} \\ &= \arg\min_j \sum_{k=1}^{F} \frac{\delta f_k}{\delta x^{(j)}} \frac{\delta L}{\delta f_k} \\ &= \arg\min_j \sum_{k=1}^{F} w_{j,k} \frac{\delta L}{\delta f_k}, \end{aligned} \tag{2}$$

where L is the loss function from the classifier, $\frac{\delta L}{\delta f_k}$ is the gradient of the loss function for the k-th feature of a word, $w_{j,k}$ is the weight in the embedding layer neural network connecting the j-th word to its k-th feature in embedding space. x^* is then perturbed by replacing it with $\tilde{x}^{(j)}$, the word from the vocabulary that has the smallest positive gradient of loss function, and, consequently, has the least influence in changing the label determined by the classifier. Mathematically, $\tilde{x}^{(j)}$ is given by:

$$\tilde{x}^{(j)} = \arg\min_j \sum_k w_{j,k} \frac{\delta L}{\delta f_k}, \text{ s.t. } \frac{\delta L}{\delta f_k} > 0 \tag{3}$$

The general schematic of the white-box gradient-based perturbation is shown in Figure 1.

Black-box Perturbation In contrast to white-box methods, black-box methods select the perturbation position randomly within unperturbed text. Consequently, black-box methods could yield adversarial examples that are less effective in misguiding the classifier than white-box generated examples. Below, we describe two black-box methods used to perturb text.

- **Black-box, Budget-Limited Perturbation (Anchor Points).** The anchor points (AP) method, proposed in (Sethi and Kantardzic 2018), is prescribed when the adversary has a limited budget and can send fewer queries to the classifier. The adversary starts with a set of valid, unperturbed samples drawn randomly from the unperturbed dataset. AP adversarial data generation proceeds in two stages called exploration and exploitation, each with its respective budget. In the exploration stage, the adversary uses one unit of exploration budget to randomly select one sample from the unperturbed set and and adds to it a perturbation vector drawn from a normal distribution $\mathcal{N}(0, R)$, where $R \in [R_{min}, R_{max})$ is a perturbation radius. The perturbed sample is sent as a query to the classifier and if it is categorized with the same label as the original sample, it is retained as a candidate adversarial example. The perturbation radius is adjusted proportional to the fraction of candidate adversarial examples and these steps are repeated until the exploration budget is expended. During the exploit phase, the adversary creates an adversarial example by generating a convex combination of a pair of randomly selected candidate adversarial examples created during exploration phase. Generating each adversarial example consumes one unit of adversary's exploitation budget.
- **Black-box, Budget-Lenient Peturbation (Reverse Engineering).** The reverse engineering (RE) attack, also proposed in (Sethi and Kantardzic 2018), is accomplished once again in two stages called exploration and exploitation. The adversary once again starts with a set of valid, unperturbed samples drawn randomly from the unperturbed dataset. In the exploration stage, the adversary trains its own stand-in classifier representing the real classifier. To get training data for its stand-in, it generates sample adversarial data by generating a random vector in a direction that is orthogonal to the difference vector between a pair of valid samples taken from the unperturbed set. The adversarial example is generated by adding the random vector and the average of the two valid samples used to generate the random vector. The adversarial example and its category obtained by sending the example as a query to the classifier, are recorded as training data and used to train the adversary's stand-in classifier. In the exploitation stage, the stand-in classifier is used to generate samples that are predicted by it to produce a desired classification. The anchor points method described above could be used for the exploitation stage. The reader is referred to (Sethi and Kantardzic 2018) further details about the AP and RE algorithms.

Gray-box Perturbation Despite their limitation of generating less effective adversarial text, black-box methods can generate larger and more diverse set of adversarial examples with fewer queries or probes to the classifier than white-box methods. With this insight, we propose to combine white-box and black-box methods into a gray-box method to investigate perturbation methods that can generate effective yet diverse adversarial examples. In the gray-box method, instead of drawing unperturbed samples for the black-box methods randomly from the original dataset, we first use the white-box method to generate a small seed set of perturbed examples from samples that are randomly-drawn from the original dataset. This seed set of perturbed examples is then used to create a larger set of perturbed examples using the AP and RE black-box methods.

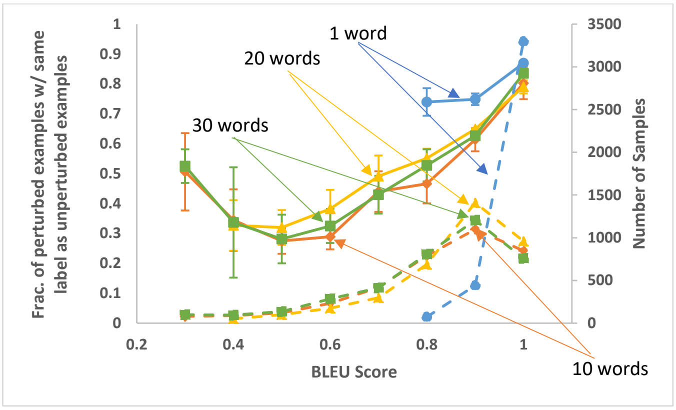


Figure 2: Variation of the ratio of correct classifications for different values of BLEU score of the perturbed text.

Experimental Evaluation

Word Embedding and Classifier Network. We have evaluated our proposed techniques on the DBPedia dataset. The dataset contains $700,000$ short articles extracted from Wikipedia and categorized into 14 different categories. For generating word embeddings from English words we have used a widely used vector representation format called Word2Vec (Mikolov et al. 2013). Word2Vec trains a neural network on a vocabulary of $150,000$ English words, each word in the vocabulary is given a unique one-hot identifier. Word2Vec gives a feature vector in a 300-dimension space for each word in the vocabulary. We have used a publicly available, pre-trained Word2Vec model for generating word embeddings for our experiments. Following recent research (Yu et al. 2017) that showed that the prefix of a long piece of text is important for determining its label, we have assumed each query is limited to the first 40 words in the text. Before sending a query text to the classifier, words in the text are given a unique integer identifier, words not in the vocabulary are given an id of zero. The classifier used for our experiments is based on (Zhang and Wallace 2017)[1]. It uses a deep neural network with three convolutional layers with filter sizes of 3, 4, and 5 respectively. A rectified linear unit

[1]Code available at https://github.com/dennybritz/cnn-text-classification-tf

(ReLU) activation function and max pooling are used with each layer. The pooled outputs are combined by concatenating them followed by a dropout layer with keep probability 0.5. The output of the dropout layer is a probability distribution indicating the confidence level of the classifier for the query text over the different data categories. The category with the maximum confidence level is output as the category of the query text. The loss function used for calculating gradients of the inupt is implemented using softmax cross entropy. The gradients are backpropagated through the classifier's network to calculate the gradients of the different features at the input. The feature gradients are again backpropagated through the Word2Vec network to calculate the gradients of the words, as shown in Figure 1.

For evaluating the effectiveness of the perturbation technique we have used the Bilingual Evaluation Understudy (BLEU) score (Papineni et al. 2002). BLEU is a widely used evaluation metric for automated machine translation of text and has recently been used to measure the difference between adversarially generated synthetic text with original text (Zhang, Xu, and Li 2017). It compares the similarity between a machine translation of text and a professional human translation without considering grammar or whether the text makes sense. We have used BLEU-4 that scores sets of four consecutive words, or 4-grams. When two pieces of text are identical their BLEU score is 1.0, and as the dissimilarity increases, the BLEU score approaches 0.

In our first experiment, we analyzed the effect of the amount of perturbation of different text samples measured in terms of BLEU score (x-axis) on the number of samples that are assigned the same label before and after perturbation (y-axis). Perturbed text was generated using the white-box gradient-based method where the features of the words with the most negative gradients, calculated using Equation 2, were perturbed with a small random amount. The number of words to perturb was varied over $\{1, 10, 20, 30\}$. For each of these perturbation amounts, a batch of 1000 text samples were perturbed, and results were averaged over 4 batches. BLEU scores of perturbed text was binned into intervals of 0.1 by rounding each BLEU score to the first place of decimal. Results shown in Figure 2, illustrate that as the BLEU score decreases (perturbed text becomes more different from the original text), the fraction of samples that retain the same label before and after perturbation also decreases. This result appears intuitive because the more different a piece of text is from its original, unperturbed version, the less likely it is to retain the same label. However, for BLUE score of 0.4 and lower, we observe that the fraction of samples retaining the same label as the original slightly increases when 10 or more words are perturbed. This is possibly due to the fact that when the perturbed text is very different from the original text, most of the perturbed words are out of context from the original text and the text appears as nonsense to a human reader. The machine classifier however is confounded into labeling the nonsensical, perturbed text with the same label as the original text, possibly due to one or more of the unperturbed words. Further investigation into this issue would lead to a better understanding of the degree of perturbation that converts text into rubbish for a human and possibly unintelligible for the machine classifier as well.

For our next set of experiments we analyzed the effect of the main algorithm parameters in the AP and RE algorithms, the maximum and minimum perturbation radii, R_{min} and R_{max} on the amount of perturbation measured in terms of BLEU score, and the fraction of samples that retain the same label as the original. Results are shown in Figures 3 through 5. We observe that for both AP (Figure 3(a)) and RE (Figure 4(a)) algorithms, as the perturbation radius increases, the BLEU score reduces, which corroborates the fact that the degree of perturbation increases the divergence between the original and perturbed text. Correspondingly, the fraction of perturbed text that retains the same label as the original text generally decreases with the increase in perturbation radius for both AP and RE, in Figure 3(b) and Figure 4(b) respectively. Figure 5 shows the BLEU score and fraction of samples that retain same label as the original versus the number of words perturbed using the white-box gradient-based approach, where the words with highest negative gradient were replaced with words with smallest positive gradient calculated using Equations 2 and 3. As before, we observe that more perturbation results in lower BLEU scores implying more divergent text after pertubation. More perturbation also reduces the fraction of perturbed samples that retain the same label as the original text.

Conclusions and Future Work

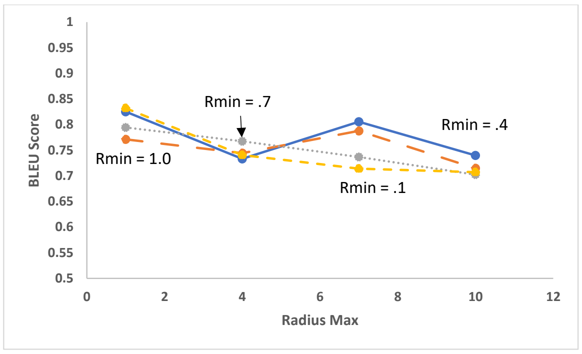
In this paper we investigated gray-box techniques for generating adversarial text as a combination of white-box gradient-based techniques and black-box techniques. We validated the correct behavior of the gray-box techniques in generating perturbed text while showing that more perturbation results in greater divergence as well as greater degree of label-changes in the perturbed text with respect to the original text. In the future, we plan to compare the proposed gray-box adversarial text generation methods with GAN-based and RNN-based synthetic text generation. While single words are usually treated as the basic lexical unit, it would be interesting to analyze how character-level perturbations and word sequence or n-gram-level perturbations affect adversarial text generation. We are also interested in getting a better understanding of how to determine a critical or minimal amount of perturbation that would be successful in generating adversarial text. We envisage that this work and its extensions will generate interesting results that would enable a better understanding and novel means for adversarial text generation, and methods to make machine classifiers robust against adversarial text-based attacks.

References

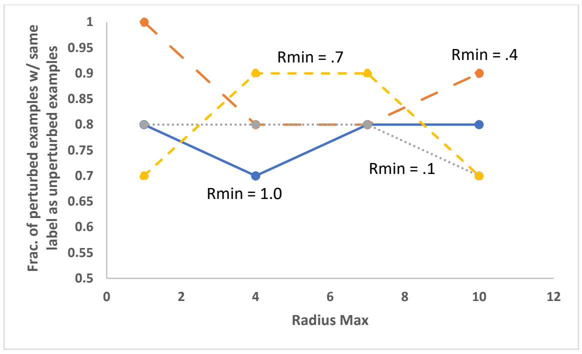
Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Bengio, Y.; Yao, L.; Alain, G.; and Vincent, P. 2013. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, 899–907.

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent

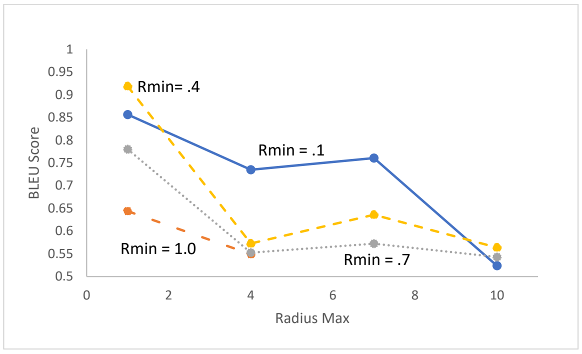


(a)

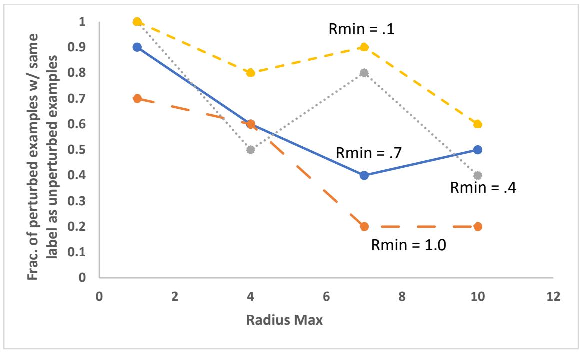


(b)

Figure 3: Variation of BLEU score (left) and fraction of perturbed examples that have same label as unperturbed text (right) for different values of R_{min} and R_{max} using anchor points (AP) algorithm.

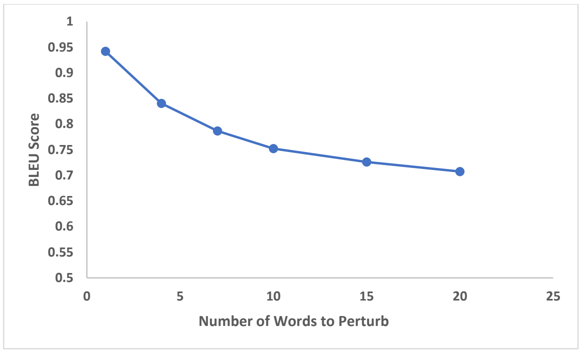


(a)

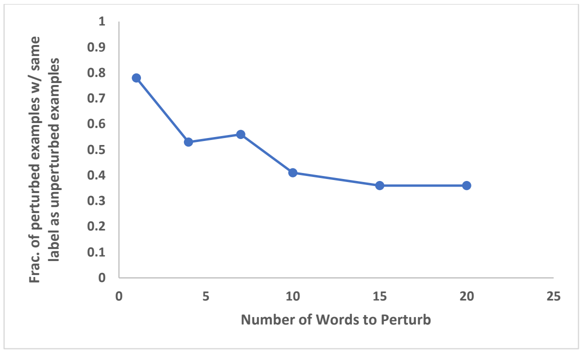


(b)

Figure 4: Variation of BLEU score (left) and fraction of perturbed examples that have same label as unperturbed text (right) for different values of R_{min} and R_{max} using reverse engineering (RE) algorithm.



(a)



(b)

Figure 5: Variation of BLEU score (top) and fraction of perturbed examples that have same label as unperturbed text (bottom) for different values of R_{min} and R_{max} using white-box gradient-based approach.

neural networks. In *Advances in Neural Information Processing Systems*, 1171–1179.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2172–2180.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, 31–36.

Fedus, W.; Goodfellow, I.; and Dai, A. M. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *arXiv preprint arXiv:1801.04354*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*, 2672–2680. Curran Associates, Inc.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *CoRR* abs/1412.6572.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; and Wierstra, D. 2015. Draw: A recurrent neural network for image generation. In Bach, F., and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1462–1471. Lille, France: PMLR.

Guo, J.; Lu, S.; Cai, H.; Zhang, W.; Yu, Y.; and Wang, J. 2017. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*.

Huszár, F. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Iyyer, M.; Wieting, J.; Gimpel, K.; and Zettlemoyer, L. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Jia, R., and Liang, P. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of Tricks for Efficient Text Classification. *arXiv:1607.01759 [cs]*.

Liang, B.; Li, H.; Su, M.; Bian, P.; Li, X.; and Shi, W. 2018. Deep text classification can be fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 4208–4215. International Joint Conferences on Artificial Intelligence Organization.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*, 3111–3119. Curran Associates, Inc.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 372–387. IEEE.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 311–318. Stroudsburg, PA, USA: Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global Vectors for Word Representation. 1532–1543. Association for Computational Linguistics.

Sethi, T. S., and Kantardzic, M. 2018. Data driven exploratory attacks on black box classifiers in adversarial domains. *Neurocomputing* 289:129–143.

Shen, T.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, 6830–6841.

Subramanian, S.; Rajeswar, S.; Dutil, F.; Pal, C.; and Courville, A. 2017. Adversarial generation of natural language. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 241–251.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2852–2858.

Zhang, Y., and Wallace, B. 2017. A sensitivity analysis of (and practitioners guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, 253–263.

Zhang, Y.; Gan, Z.; Fan, K.; Chen, Z.; Henao, R.; Shen, D.; and Carin, L. 2017. Adversarial feature matching for text generation. In *International Conference on Machine Learning*, 4006–4015.

Zhang, H.; Xu, T.; and Li, H. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, 5908–5916.